Relatório - Projeto Intermediário/Relógio

Matheus Castellucci e Rodrigo Medeiros

1 - Arquitetura Escolhida

Ao contrário da última parte do projeto, esta versão foi implementada utilizando a arquitetura registrador-memória.

Nessa abordagem, um banco de registradores está disponível para realizar as operações da Unidade Lógica Aritmética (ULA), o que a torna uma das mais rápidas. No projeto do circuito, simplesmente adicionamos o banco de registradores logo antes da entrada da ULA, mais especificamente na entrada B.

A adoção dessa nova arquitetura resultou em mudanças na forma como as instruções operam; agora, temos 2 bits adicionais para decodificar, totalizando 12 bits no endereço do bloco de registradores.

2 - Total de instruções e seus pontos de controle:

A tabela abaixo resume o total de instruções e sua sintaxe. O projeto foi desenvolvido com as instruções já passadas em sala, ou seja, nenhuma instrução nova foi criada. Como não adicionamos nenhuma instrução nova, não precisamos de pontos de controle novos.

OBS: Passa = 10, Soma = 01, Sub = 00

Instrução	Mmnemônico	Hab Escrita Retorno	JMP	RET	JSR	JEQ	Sel Mux	Hab _A	Operação	habFlag=	RD	WR
Sem operação	NOP	0	0	0	0	0	X	0	XX	0	0	0
Carrega valor da memória para A	LDA	0	0	0	0	0	0	1	10	0	1	0
Soma A e B e armazena em A	SOMA	0	0	0	0	0	0	1	01	0	1	0
Subtrai A e B e armazena em A	SUB	0	0	0	0	0	0	1	00	0	1	0
Carrega valor imediato para A	LDI	0	0	0	0	0	1	1	10	0	0	0
Salva valor do A para a memória	STA	0	0	0	0	0	0	0	XX	0	0	1
Desvio de execução	JMP	0	1	0	0	0	X	0	XX	0	0	0
Desvio condicional de execução	JEQ	0	0	0	0	1	X	0	XX	0	0	0
Comparação	CEQ	0	0	0	0	0	0	0	00	1	1	0
Chamada de Sub Rotina	JSR	1	0	0	1	0	0	0	XX	0	0	0
Retorno de Sub Rotina	RET	0	0	1	0	0	0	0	XX	0	0	0

Tabela 1: Instruções, e seus pontos de controle correspondentes, utilizadas no projeto

3 – Formato das instruções

Cada instrução possuirá o seguinte formato:

- 15 bits no total
  - 4 de *opcode*
  - *2 de endereçamento de registrador*
  - 9 de endereçamento/imediato

O opcode funciona como um “código”, de 4 bits, que o decodificador utiliza para identificar qual instrução deve executar. Cada uma das 12 instruções listadas na Tabela 1possui o seu próprio opcode. Caso ocorra algum erro no opcode, ou seja, se chegar um opcode que não exista, a instrução retornada será um “NOP”.

Instrução	Opcode
NOP	0000
LDA	0001
SOMA	0010
SUB	0011
LDI	0100
STA	0101
JMP	0110
JEQ	0111
CEQ	1000
JSR	1001
RET	1010

Tabela 2: Instruções e seus respectivos opcodes.

No endereçamento, utilizamos 9 bits para representar o endereço desejado, que varia de 0 a 511 em binário.

Quando desejamos enviar um valor imediato, o nono bit é obrigatoriamente 0, o que significa que os valores variam de 0 a 255.

4 – Fluxo de dados do processador

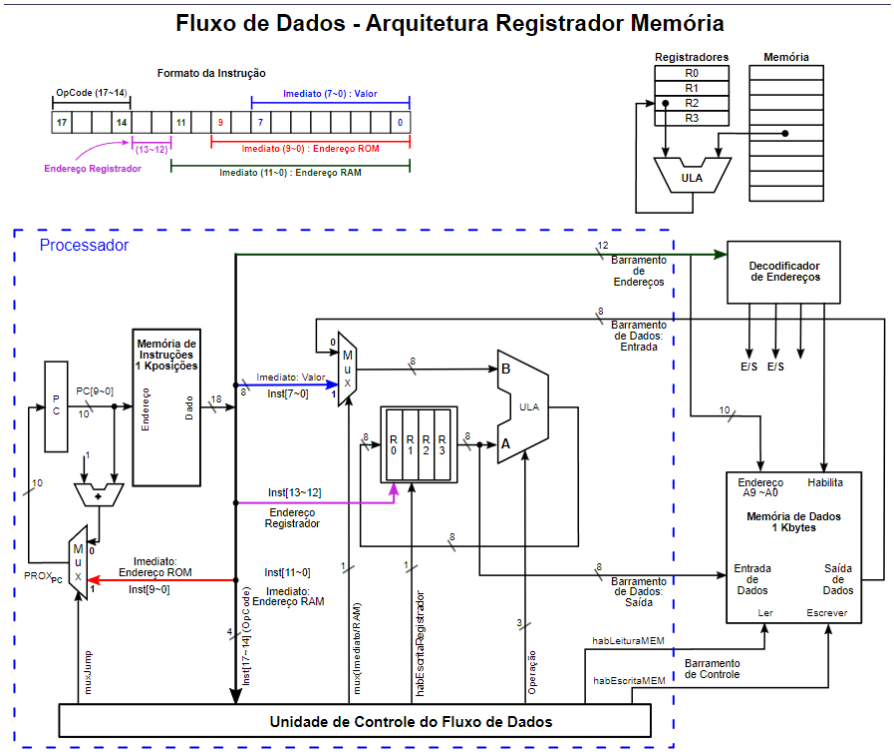

Imagem 1: Circuito que representa o fluxo de dados do processador

A Figura 1 ilustra o fluxo de dados da arquitetura "registrador-memória". Conectado à saída da ULA, há um banco de registradores na entrada A. Esses registradores podem ser usados para armazenar valores utilizados nos cálculos da ULA. Também é possível obter um valor da memória e utilizá-lo na entrada B, ou simplesmente armazená-lo nos registradores do banco.

5 – Pontos de controle e suas utilizações

Cada ponto de controle representa uma instrução para o processador fazer. Abaixo uma descrição mais detalhada de cada uma:

- NOP: Sem operação
- LDA: Carrega o valor presente na posição da memória desejada e salva no acumulador
- SOMA: Soma o valor presente na posição da memória desejada e salva no acumulador. Pode somar também um valor imediato
- SUB: Subtrai o valor presente na posição da memória desejada e salva no acumulador. Pode subtrair também um valor imediato
- LDI: Carrega um valor imediato e salva no acumulador

- STA: Salva o valor do acumulador na posição da memória desejada.
- JMP: Desvio de execução, ou seja, pula para a linha desejada
- JEQ: Desvio de execução condicional, ou seja, pula para a linha desejada se o valor do acumulador for igual o valor da posição de memória desejada.
- CEQ: Comparação, ou seja, compara o valor do acumulador com o valor da posição de memória desejada. Serve como uma espécie de “resposta” para o JEQ.
- JSR: Desvio para subrotina, ou seja, pula para a linha desejada e executa até ser requisitado o retorno.
- RET: Retorno da subrotina, ou seja, volta para 1 linha depois que foi executado o JSR.

6 – Rascunho do diagrama de conexão do processador com os periféricos

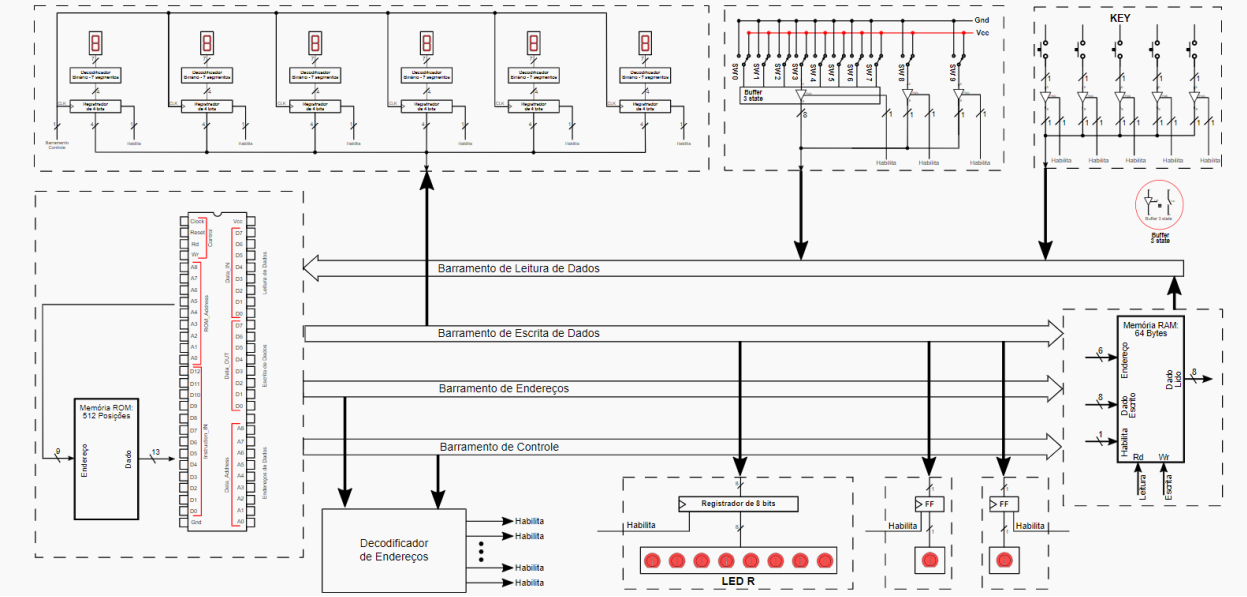

Imagem 2: Diagrama de conexão do processador com os periféricos (chaves, botões, LEDRs e Displays)

A conexão entre o processador e os periféricos, conforme ilustrado na Imagem 2, é feita por meio de dois decodificadores que recebem 3 bits e retornam 8. Dependendo da saída do decodificador 2, que representa o endereço, uma porta AND correspondente será ativada, acessando o periférico desejado. As chaves e os botões estão conectados ao barramento de leitura, ou seja, eles escrevem no bit menos significativo desse barramento. Por outro lado, os displays e os LEDs estão no barramento de escrita.

7 – Manual de uso

Após compilar o programa, já é possível utilizar a placa como um relógio. Aqui estão as coisas que você precisa saber:

- Ao compilar o programa na placa, ela começará a contagem do tempo imediatamente sem precisar apertar nada.
- O código possui um modo de contagem de tempo normal, em que 1 segundo de fato equivale á 1 segundo, e outro modo em que a contagem de tempo é 1000 vezes mais rápida. Para alterar entre os modos, basta mexer no SW9 (vale mencionar que não há problema em ficar alterando a base sem parar):
    - SW9 = 0: Base normal de tempo.
    - SW9 = 1: Base rápida de tempo.
- Caso queira-se recomeçar a contagem de tempo, o botão FPGA_RESET foi configurado, para assim como seu nome indica, resetar a contagem. Ao apertá-lo, todos os displays Hexadecimais voltarão a zero.
- Ao chegarmos em 24 horas, o relógio irá sozinho se resetar, ou seja, quando estamos em 23h59min59 e o segundo passa, voltaremos para 00h00min00 automaticamente.
- O relógio também possui a funcionalidade de estabelecer o tempo no qual se começara a contagem. Para isso precisamos seguir os seguintes passos:
    - Passo 1: Com o relógio já funcionando, devemos apertar o KEY1 para entrar no modo de colocar o tempo.
    - Passo 2: Logo depois de apertar o KEY1, entramos no modo de incrementar a hora. Cada vez que o botão KEY0 for apertado nesse modo é traduzido no incremento de uma hora. A hora máxima é 23, se KEY0 for apertado depois disso, a hora retorna para 0.
    - Passo 3: Depois de ter a hora desejada, devemos apertar o KEY1 de novo, isso nos leva para o modo de incrementar os minutos. Funcionando do mesmo jeito, apertamos o KEY0 para realizar incrementos unitários, e ao chegar ao limite superior de 59 minutos, ele volta para 0 minutos.
    - Passo 4: Com os minutos já também definidos, apertamos KEY1 de novo, nos levando para o modo de incrementar segundo que funciona do mesmo que o anterior. Cada aperto de KEY0 leva a um incremento unitário de segundo, com um máximo de 59 segundos e voltando para zero depois disso.
    - Passo 5: Agora com o horário desejado já configurado, basta apertar KEY1 uma última vez, fazendo com que volte a se contar o tempo a partir do desejado.
- Por último, também podemos pausar o relógio ao apertar KEY0, enquanto nesse estado, subsequentes apertos de KEY0 geram incrementos unitários nos segundos. Para sair desse modo, basta apertar KEY1.